**İ.T.Ü.**

**Faculty of Computer and Informatics**

**Computer Engineering**



**MICROCOMPUTER LAB**

**REPORT**

|  |  |  |
| --- | --- | --- |
| **Lab No** | **:** 03 | |
| **Lab Date** | **:** 10.10.2013 | |
| **Group** | **:** B9 | |
| **Group Members** | **:** 040100014 Teoman TURAN  040100018 Mustafa DURMUŞ  040100117 Tuğrul YATAĞAN  040100124 Emre GÖKREM | |
|  |  | |
|  |  | |
|  |  | |
| **Research Assistant :** Cumali TÜRKMENOĞLU | | |  |
| **1. THE AIM/CONTENT of THE EXPERIMENT** | |  |

The purpose of doing this experiment is examining relationship between calling subroutine and stack pointer on Motorola 6802.

**2. EQUIPMENT**

Only ITU-Training Kit has been used in the experiment. This kit consists of the following hardware components:

* CPU: MC6802
* Memory: 24K\*8 R/W + 16K\*8 Read Only
* Address decoder
* Control unit
* Display and Keypad
* Parallel Port
* Serial Port
* Programmable counter

In addition to this kit, for observing the simulation results during the experiment, one of our group members has used his own computer to run the simulation software of the microprocessor.

**3. THE PROGRAMS for THE EXPERIMENT**

The machine codes of the instructions in the code fragment below and their addressing modes had been determined before coming to the lab. Also, that code fragment had been written and run on the simulator software of the microprocessor. In the lab, the code has been run on ITU-Training Kit. Here are the machine codes of the instructions, their addressing modes, what they do and screenshots for each of them from the simulation software of the microprocessor:

**3.1. SECTION 2: CALLING SUBROUTINES**

The following calling subroutine code is written and then converted the assembly into machine code and run on ITU-TRAINING Kit:

lds #$5f00

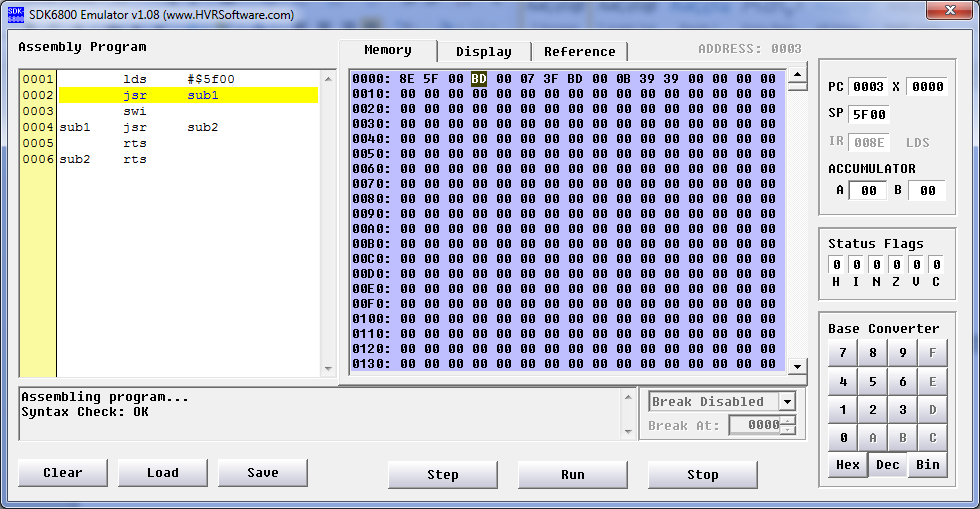
jsr sub1

swi

sub1 jsr sub2

rts

sub2 rts



It is shown below in the table which is include stack pointer and its values after each instruction is executed.

|  |  |  |
| --- | --- | --- |
| **PC** | **STACK\_POINTER** | **Memory[STACK\_POINTER]** |
| 0000 | F000 | 00 |
| 0003 | 5F00 | 00 |
| 0007 | 5EFE | 00 |
| 000B | 5EFC | 00 |
| 000A | 5EFE | 0A |
| 0006 | 5F00 | 06 |
| 0000 | 5EF9 | 00 |
| 0003 | 5F00 | 00 |

**3.2. SECTION 2: ARGUMENT PASSING**

The following argument passing code is written and then converted the assembly into machine code and run on ITU-TRAINING Kit:

lds #$5f00

ldaa #$23

psha

jsr sub

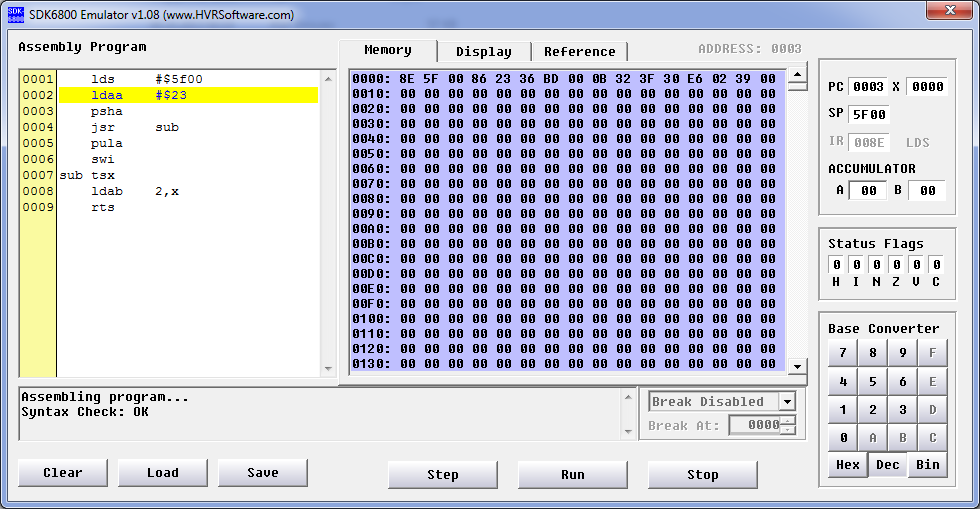
pula

swi

sub tsx

ldab 2,x

rts



It is shown below in the table which is including stack pointer and its values after each instruction is executed.

|  |  |  |
| --- | --- | --- |
| **PC** | **STACK\_POINTER** | **Memory[STACK\_POINTER]** |
| 0000 | F000 | 00 |
| 0003 | 5F00 | 00 |
| 0005 | 5F00 | 00 |
| 0006 | 5EFF | 00 |
| 000B | 5EFD | 00 |
| 000C | 5EFD | 00 |
| 000E | 5EFD | 00 |
| 0009 | 5EFF | 09 |
| 000A | 5F00 | 23 |
| 0000 | 5EF9 | 00 |
| 0003 | 5F00 | 00 |

**3.3. SECTION 3: EXPERIMENT**

Here is a program which calculates i’th value of Fibonacci numbers. This program starts from the address $4000 and applies the following algorithm:

int fibonacci(int n){

if(n==1 || n==0)

return n;

else

return (fibonacci(n-1)+fibonacci(n-2));

}

Assembly code of algorithm is converted from algorithm. Here is the code of the program in Motorola 6800 Assembly programming language:

n .equ #6 // value for calculate

ldx #$0050

lds #$0080

ldab #n

sub1 cmpb #0

ble topla

cmpb #1

ble topla

sub3 decb

pshb

decb

pshb

sub4 pulb

sts $0050

cpx 0081

bge son

jmp sub1

topla INC $0090

jmp sub4

son swi

It has been converted into machine codes and run on ITU-Training Kit. Here are screenshots demonstrating the result from the simulation software of the microprocessor:

